# Visual Interfaces for the Development of Event-based Web Agents in the IRobot System

Liangyou Chen

ACM Member

chen_liangyou@yahoo.com

**Abstract.** Timely integration and analysis of information from the World-Wide Web is important for businesses and startups, especially to cope with the increasing global competition need. Manual Web data analysis is time consuming and error prone, while developing automated algorithms is nontrivial even for expert Web programmers. We argue that an event-based architecture that couples Web-agent technology with visual-programming methodology is ideal for Web-data analysis. We demonstrate this with the IRobot system, which uses visual interfaces to facilitate the design of event-based intelligent Web agents for data integration. The system is effective in both its ability to capture human-oriented Web procedures and its flexibility in modeling complex agents that meet human goals.

**Keywords:** Web computing, rule-based systems, Web agents, data integration, Web automation

## 1 Introduction

The Word-Wide Web has grown as a central platform for publishing and distributing information. Small businesses and startups are more and more dependent on information from the Web, e.g., for price comparison, market analysis, real-time trend discovery, literature research, and many other activities. A real problem for them is the timely integration and analysis of information from multiple Web resources. Because of time and financial constraints, they are unable to hire special Web programmers to develop complex systems. Rather, they depend much on ad hoc and manual approaches to pull information around the Web, which is both labor intensive and time consuming.

There are efforts from both the research and the industrial communities trying to address this issue. The research community proposes the use of intelligent agent systems to support autonomous data integration from the Web, e.g., Michalowski *et al.* [1] and Neiling *et al.* [2]. Industrial solutions typically use special libraries in a hosting programming language for Web automation. Examples include cURL [3] in PHP, and Scrapy [4] in Python. Utilizing these technologies still requires substantial knowledge and effort in programming and system building. Also, many Web resources publish standard Web services or application programming interfaces for

third-party application integration, which should allow automated integration and processing of information from their sites through simple scripts. However, many other websites publish only traditional Web pages mainly for human exploration, and integration and analysis of information from such websites is especially difficult for non-experts.

Our goal here is to offer an affordable, easy to use software platform for small businesses to integrate and process available Web resources without requiring much programming skill. We developed a solution through the use of event-based Web agents, or robots, with the IRobot (standing for Internet Robots) system, which is able to simulate user's navigation on the Web while performing data retrieval and manipulation. The system can be learned and mastered by non-experts because of the use of visual-programming interfaces. The system has attracted thousands of registered users worldwide, and received favorable feedback from a public Web forum.

## 2 A Motivating Example

To recommend the best doctor for patients, Mary, a medical consultant, was considering the use of Google Scholar citation index [5][6] to evaluate the expertise of medical doctors. Her idea was that doctors who published more quality papers should be more knowledgeable about the medical condition than their peers, and should be able to provide better treatments. The citation index provided by Google Scholar can serve as a reference for the quality and impact of the doctor's publications. Mary was thinking of the following procedure:

**Procedure 1: Google Citation Index**

- Get the disease name of a patient from a database;
- Find from Google Scholar [5] publications related to the disease;
- Get the author names of each publication;
- Go to Google Citation Gadget [6] for citation index;
- Find the citation index score for each author;
- Rank authors based on their citation-index scores.

Mary found IRobot software from the Internet. After spending a few hours to get familiar with the system, she quickly created a robot to complete the above procedure. Unfortunately, she found out that Google Scholar citation index only uses the last name and first-name initial of the author to search, and popular names such as "J. Smith" has a much higher score than rare names, because there are many J. Smiths in the world. She decided to use another Web service named "Scholarometer" [7], which is a crowd-source scholar-rating system provided by Indiana University, to rate scholars, and use Pubmed [8] to find disease-related publications. She changed the procedure as follows:

Procedure 2: Scholarometer Impact Score

- Get the disease name of a patient from a database;
- Find from Pubmed [8] publications related to the disease;
- Get the author names of each publication;
- Use Scholarometer Web service [7] at Indiana University to search for the author's impact score;
- Find the impact score for each author;
- Rank authors based on their impact scores.

Now, Mary can easily make a robot for Procedure 2. She found that the new impact scores are more reliable than the citation index score. Based on the ranking, she had more confidence when recommending doctors to her patients.

In this example, Mary can test her idea directly on the Web using IRobot software. She can continuously change and refine her idea without much cost. The resulting robot improved the quality of her work. (Note that although the above two procedures look similar, it requires significant programming effort to retrieve and locate Web data at different websites using regular programming techniques.)

## 3 The IRobot System

IRobot is a system for the design and deployment of Web agents for Web data processing. The system includes convenient visual-programming interfaces for the composition and combination of high-level Web and database operations into actionable software agents. The creation and operation of agents can be easily followed and mastered by casual users. Also, the system provides a full range of lower-level data-operation functions, such as text transformation, date and time operations, and logic computation, for skilled users. An event-driven architecture is employed for the integration of features and functions at different levels.

Internally, IRobot uses a specially-designed XML-based rule language to represent Web and database operations, which are termed "actions" in the IRobot system. For example, each step in Procedure 1 and Procedure 2 can be seen as an action in the IRobot system, and can be represented as an "Action" tag in XML. The benefit of an XML-based rule language is that rules can be easily composed and manipulated by software. This, for example, allows the IRobot software to automatically learn action rules from user-Web interactions, and encode them in XML.

In IRobot, a sequence of actions comprises a "task," and a robot may include multiple tasks. Each robot is stored and maintained in a single robot file, and represents a logically complete job. In our example, Procedure 1 and 2 can be designed as two tasks named "Google Citation Index" and "Scholarometer Impact Score," and they can be included in a single robot, named "scholar_index." In addition, we allow users to flexibly divide a task into multiple smaller tasks and combine them by "task calls." The concepts of robot, task, and action also allow users to visualize Web interactions as objects of different granularities, and provide a means to solve complex problems by visually composing and combining simpler

operations. A more complete discussion of features of the IRobot system can be found from our online manual at http://irobotsoft.com/help/.

Fig. 1 shows the main interface of IRobot. The interface shows user-designed robots on the right in an embedded Internet Explorer (IE) Web browser, and the list of robot actions on the left panel. Each action can be customized as an object. A user can open and run robots from the main interface by simple clicking. When a robot is running, it will show the real-time Web interactions in the embedded Web browser.

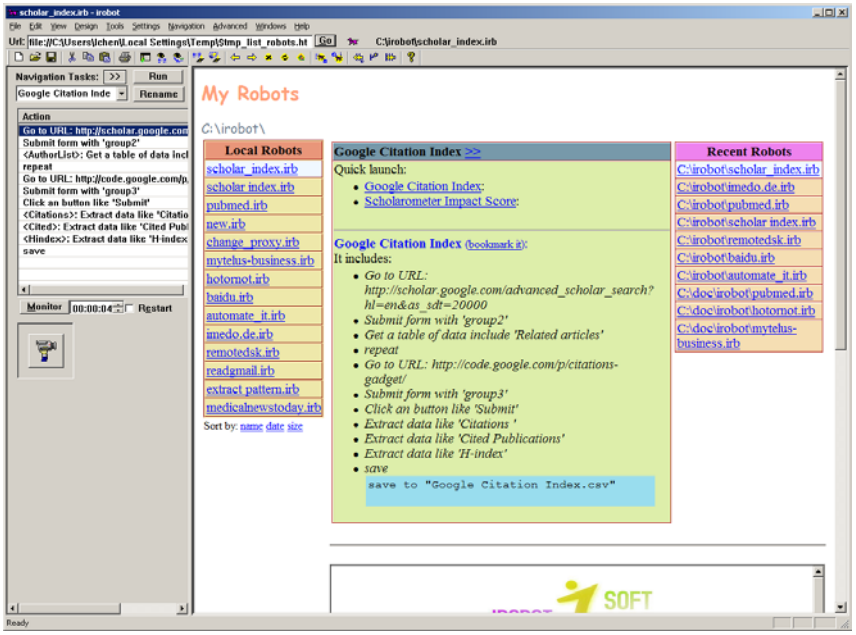


**Fig. 1.** The main interface of the IRobot system. On the right it uses an embedded IE Web browser to list user-designed robots. On the left, it shows the actions of a selected robot.

### 3.1 Visual-Programming Interface

IRobot allows users to visually compose and combine actions representing Web and database operations. Actions can be created in IRobot by simple recording. Specifically, IRobot provides a recorder-like interface, which automatically generates a sequence of robot actions when the user navigates in the embedded Web browser. These actions can be used to repeat what the user has done in the browser, such as link following, input feeding, form submission, or data extraction. More importantly, these actions are resistant to Web page changes and can work continuously on dynamically generated Web content. Internally, we use robust wrapper techniques reported in [9] and [10] to locate Web data.

Once the actions are generated, the user can move them around in an object oriented fashion, or customize their properties via a Web-browser based interface. Fig. 2 shows the customization of the properties of a “Get a table of data” action,

which is given a name "AuthorList." Customizable properties include the location of the data, the sequential order for retrieving each tuple of data, the text description of the action, and so on. Notice that these properties were set initially by the recorder, and typically users simply need to choose another option to change the action's behavior.

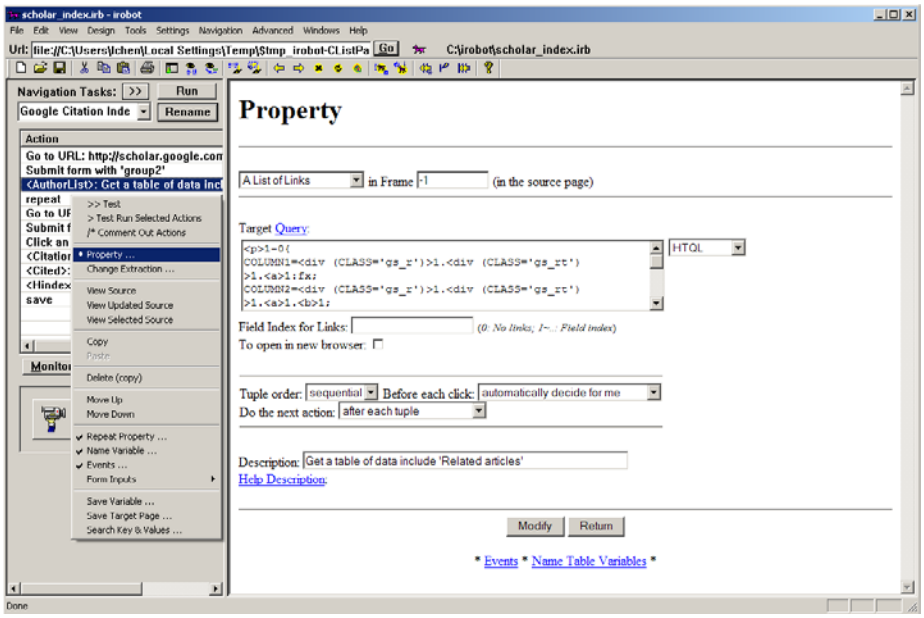


**Fig. 2.** Property of a robot action that has been automatically generated by the recorder.

## 3.2 Event Customization

Fine-grain control and customization of robots is done through event-condition-action (ECA) rules. For this, we have defined special events corresponding to different stages of retrieving a Web page or retrieving a tuple of results from the Web page. For example, a "before each page" event is associated with the time before a page is retrieved by the robot, and an "after each tuple" event is associated with the time after a tuple is extracted from the page. Users can then use these events to fire action rules, for example, to compute new variables, or to call robot tasks.

Fig. 3 shows the use of events in the "scholar_index" robot to transform the author names. Here, the user uses some low-level functions like "htql" and "loadData" (specifications available in our manual), and associates them with an "after each tuple" event to divide author names from each tuple of results (which was defined in variable "AuthorList"). The ECA rules serve to separate the relatively simple computations from the more complex Web related operations (i.e., data extraction and Web navigation operations).

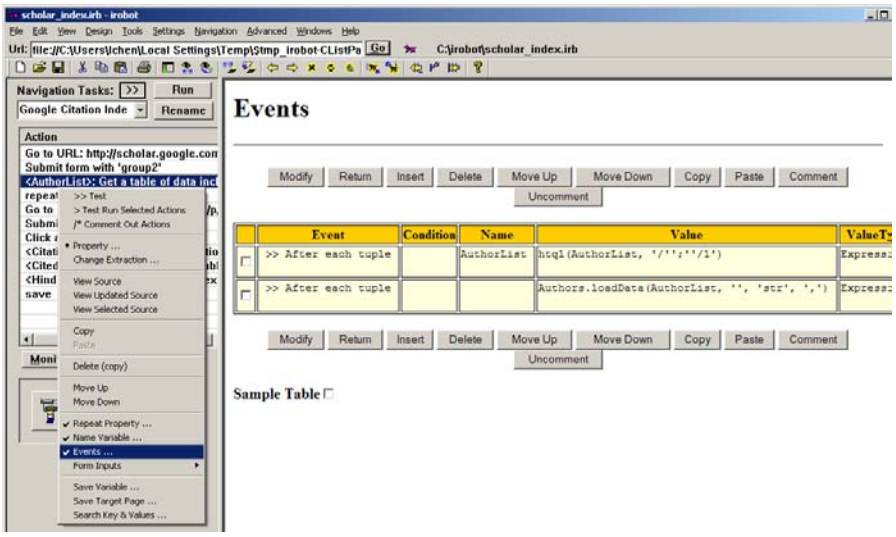


**Fig. 3.** Visual interfaces to customize events in the IRobot system.

### 3.3 Database Operations

IRobot is also a data-integration engine. It supports the integration of general data sources including most commercial databases, text files, HTML files, and XML files. Each database or file is defined as a named data source in IRobot with simple wizard-like interfaces. Once defined, they can be used to locate or save data.

Most of the database operations can be defined with visual interfaces. For example, Fig. 4 shows the interface to save and sort data in a text database in CSV (standing for Comma-Separated Values) format. The sorting fields are simply listed in the "Sorting by fields" box, and duplicated values will be removed by selecting an option from a drop-down list, e.g., in this case "Unique & Keep Old Data & Append File," which ensures that new data with the same unique keys will not be added to the text database. Finally, this database operation is associated with an "after each tuple" event, so data are automatically fed and saved to the database when each tuple of result is extracted. The combination of visual interfaces and ECA rules provides a mechanism for users to define database operations without writing complex structured query language, or SQL, statements.

## 4 How IRobot Works

As demonstrated in Fig. 1, Procedure 1 in our example is realized in IRobot as a task named "Google Citation Index," including the following sequence of actions:

*a.* Go to URL: http://scholar.google.com/...;
*b.* Submit form with 'group2';
*c.* Get a table of data including 'Related articles';

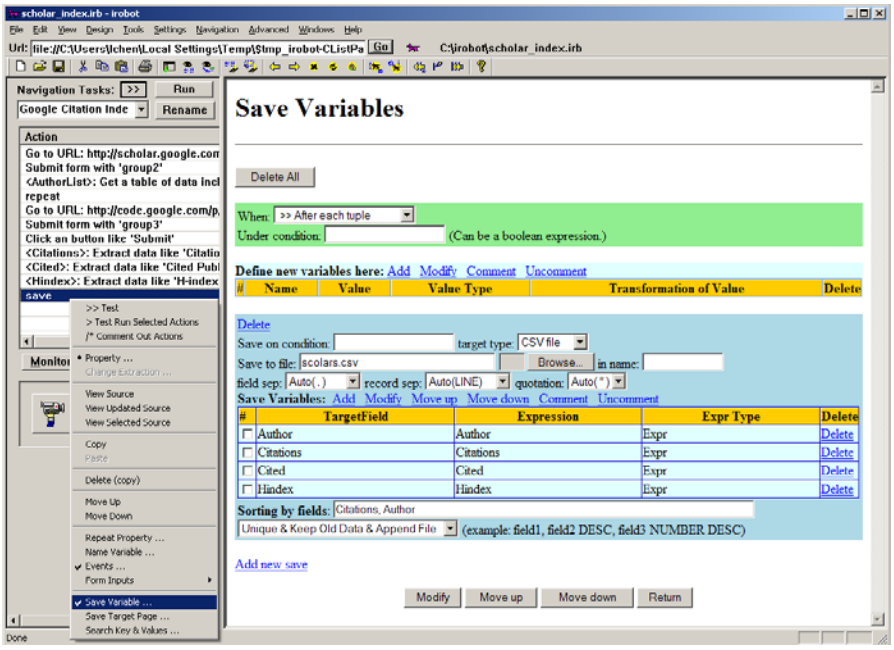


**Fig. 4.** Visual interfaces in the IRobot system to save and sort data to databases.

*d.* Repeat;
*e.* Go to URL: http://code.google.com/p/citations-gadget/;
*f.* Submit form with 'group3';
*g.* Click an button like 'Submit';
*h.* Extract data like 'Citations';
*i.* Extract data like 'Cited Publications';
*j.* Extract data like 'H-index';
*k.* Save.

This sequence of actions is very similar to the steps in Procedure 1, and a user can easily understand the workflow of this task by simply looking over the action list. However, such visual simplicity disguises much complexity in the actual performance of this task. The run-time complexity of the robot comes from two sources. First, the sequence of actions is not exactly carried out in sequence – they are performed in a recursive manner, i.e., each later action is carried out repeatedly after each tuple of data is processed by its preceding action. This recursive behavior mainly affects actions that produce multiple tuples. For example, action *c* in the above list would extract multiple "Related articles," and because of recursion, each article will be further processed by actions from *d* to *k*.

The second source of run-time complexity comes from various programming constructs including action repeating, conditional branching, and task calling. These programming constructs provide a means to fine-control the execution logic of a robot, and they can be defined visually. For example, action *d* above repeats on each author extracted from action *c*, and also due to recursion, actions *e*-*k* will be recursively applied to each author. Conditional branching and task calling are

mainly done with ECA rules, where, based on certain condition, another task may be called for execution just like function calls in regular programming languages. For example, in our software demonstration for Procedure 2, the Scholarometer Web service is designed as a separate task, and is called from the main task "Scholarometer Impact Score" after each author is found from Pubmed.

## 5 More about the IRobot System

IRobot is free software available at http://irobotsoft.com/. Video demos and detailed manuals can be found online at: http://irobotsoft.com/help/. An active discussion forum is at: http://irobotsoft.org/bb/. Our members love the software. For example, our forum member "herbycanopy" said:

> *"I must say this program really is great, you all seem to have thought of everything."*

Another recent comment from member "linkme" said:

> *"I love this software … But … When are you going to go xPlatform with it? It doesn't play as well as it could in wine :p"*

Through the use of ECA rules and visual-programming interfaces, IRobot offers great simplicity for the design of Web-data integration agents. IRobot decreased the cost of small businesses for Web-data collection and analysis. For example, one of our customers stated in email:

> *"Desperate to work with someone who is reasonably priced and that we can trust and you have never let us down."*

## References


1. Michalowski, M., Ambite, J. L., Thakkar, S., Tuchinda, R., Knoblock, C. A., Minton, S.: Retrieving and Semantically Integrating Heterogeneous Datafrom the Web. IEEE Intelligent Systems, 19, 72--79 (2004)
2. Neiling, M., Schaal, M., Schumann, M.: WrapIt: Automated Integration of Web Databases with Extensional Overlaps. Web, Web-Services, and Database Systems, 2593/2009, 184--198 (2009)
3. cURL, http://curl.haxx.se/
4. Scrapy, http://scrapy.org/
5. Google Scholar, http://scholar.google.com/
6. Google Citation Gadget, http://code.google.com/p/citations-gadget/
7. Scholarmeter, http://scholarometer.indiana.edu/
8. Pubmed, http://www.ncbi.nlm.nih.gov/pubmed/
9. Chen, L., Jamil, H. M., Wang, N.: Automatic Composite Wrapper Generation for Semi-structured Biological Data Based on Table Structure Identification. SIGMOD Record, 33, 58--64 (2004)
10. Chen, L.: Ad Hoc Integration and Querying of Heterogeneous Online Distributed Databases, Ph.D. Dissertation. Dept. of Comp. Sci. & Eng., Miss. State Univ., MS (2004)